Frequency domain volume rendering by the wavelet X-ray transform

Westenberg, Michel A.; Roerdink, Jos B.T.M.

Published in:
Ieee transactions on image processing

Publication date:
2000

Frequency Domain Volume Rendering by the Wavelet X-Ray Transform

Michel A. Westenberg, *Student Member, IEEE,* and Jos B. T. M. Roerdink, *Member, IEEE*

***Abstract*—We describe a wavelet-based X-ray rendering method in the frequency domain with a smaller time complexity than wavelet splatting. Standard Fourier volume rendering is summarized and interpolation and accuracy issues are briefly discussed. We review the implementation of the fast wavelet transform in the frequency domain. The wavelet X-ray transform is derived, and the corresponding Fourier-wavelet volume rendering algorithm (FWVR) is introduced. FWVR uses Haar or B-spline wavelets and linear or cubic spline interpolation. Various combinations are tested and compared with wavelet splatting (WS). We use medical MR and CT scan data, as well as a 3-D analytical phantom to assess the accuracy, time complexity, and memory cost of both FWVR and WS. The differences between both methods are enumerated.**

***Index Terms*—Fourier volume rendering, Fourier-wavelet volume rendering, wavelet splatting, wavelet X-ray transform.**

I. Introduction

VISUALIZATION and exploration of large three-dimensional (3-D) digital data volumes is becoming increasingly popular. Volume rendering is prominent among the techniques which have been developed for this purpose, using advanced computer graphics techniques such as illumination, shading and color. The desire to exchange volume data through systems such as the Internet has created a need for fast and efficient methods of transfer and display. To relieve the demand on the server capacity, volume data may be stored on a central server, while (part of) the rendering is performed on client systems. Not all of these clients will have a high-bandwidth network connection, so that we need a mechanism to visualize data incrementally as it arrives ("progressive refinement"). For this purpose multiresolution models are developed, allowing systematic decomposition of the data into versions at different levels of resolution. Another benefit of such approaches is local level-of-detail (LOD), i.e., using a lower resolution for small, distant or unimportant parts of the data. Such a mechanism is provided by the wavelet technique.

This paper is concerned with a direct volume rendering method [1] called *X-ray volume rendering*, which is based upon integrating the 3-D data along the line of sight, yielding a two-dimensional (2-D) image in the view plane. The method supports shading and depth-cueing [2], [3], but no occlusion and perspective projection. However, it turns out to be one of the preferred techniques for medical applications, because physicians are well-trained in interpreting X-ray like images for diagnosis. The corresponding mathematical concept is the *X-ray transform*, well-known from computerized tomography [4]. An efficient way to compute this transform makes use of frequency domain techniques [5], [6]; in the following, this method is referred to as *Fourier volume rendering*, abbreviated as FVR. After an initial 3-D Fourier transform of the data, a view direction $\boldsymbol{\theta}$ is chosen and the values of the Fourier transform in a plane, called the *slice plane*, perpendicular to $\boldsymbol{\theta}$ are computed. Interpolation in frequency space is necessary to obtain the values of the Fourier transform of the function to be visualized at a regular pixel grid in the slice plane. A subsequent inverse 2-D Fourier transform gives the desired image in the view plane. The time complexity of FVR is dominated by the 2-D inverse Fourier transform from the slice plane to the view plane, hence is $\mathcal{O}(N^2 \log N)$ for a volume data set of size $N \times N \times N$. Frequency domain volume rendering algorithms have to deal with problems related to high interpolation cost and high memory cost. However, since processing power has increased and memory costs have dropped significantly, these problems are less serious on modern hardware.

Another volume rendering method based on the X-ray transform is *splatting* [7]. This is an object order method which reconstructs a continuous function from discrete data by convolution with a reconstruction filter, followed by a mapping to the image plane by superposition of building blocks called "splats" or "footprints." This method supports occlusion, shading, and perspective projection, but suffers from color bleeding or "popping" [8] and aliasing [9]. Previously, *wavelet splatting* has been proposed by Lippert and Gross [10] as an extension of the splatting method, see also [3], [11]. Wavelet splatting modifies the splatting algorithm by using wavelets as reconstruction filters, so that data can be visualized at different levels of detail. Just as the original splatting method, the time complexity of this algorithm is $\mathcal{O}(N^3)$.

The purpose of this paper is the derivation of a wavelet-based X-ray rendering method with a smaller time complexity than wavelet splatting. Since in ordinary volume rendering this goal can be achieved by frequency domain techniques, a similar approach is followed in this paper, resulting in an algorithm with the same time complexity as ordinary FVR, i.e. $\mathcal{O}(N^2 \log N)$.

To achieve this goal, we study the *wavelet X-ray transform*, as introduced in [12], which combines integration along the line of sight with a simultaneous 2-D wavelet transform in the plane perpendicular to this line. A closely related transform was introduced in [13], [14], and combines integration over a line with a

Manuscript received May 6, 1999; revised November 30, 1999. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Kannan Ramchandran.

The authors are with the Institute for Mathematics and Computing Science, University of Groningen, 9700 AV Groningen, The Netherlands (e-mail: michel@cs.rug.nl; roe@cs.rug.nl).

Publisher Item Identifier S 1057-7149(00)05316-1.

simultaneous 1-D wavelet transform along this line. We derive an efficient implementation of the wavelet X-ray transform by using a frequency domain implementation of the wavelet transform [15], [16]. This is particularly efficient when the length of the wavelet decomposition and/or reconstruction filters is large, as is the case for some of the basic wavelets (e.g. B-spline wavelets) used in this paper. This results in an algorithm whose initial step, i.e. computation of the Fourier transform in a slice plane, is identical to that of ordinary FVR. The additional step is a wavelet decomposition of the slice plane data in Fourier space to a given level of detail. Approximation images are then obtained by a partial wavelet reconstruction in Fourier space, followed by a 2-D inverse Fourier transform. Since wavelet detail coefficients are available in Fourier space, progressive refinement is straightforward. The Fourier space representation does not allow local level-of-detail.

The organization of this paper is as follows. Section II summarizes standard Fourier volume rendering. Interpolation and accuracy issues are briefly discussed. Section III introduces some basic wavelet concepts, and reviews implementation of the wavelet transform in the frequency domain. Section IV then introduces the wavelet X-ray transform and the corresponding Fourier-wavelet volume rendering (FWVR) algorithm. A comparison of the new method and wavelet splatting with respect to accuracy, time complexity and memory cost is presented in Section V. Section VI contains a summary and discussion of future work.

II. Fourier Volume Rendering

Fourier domain volume rendering methods [5], [6] provide an implementation of X-ray volume rendering, where the volume data are integrated along the line of sight. The mathematical basis is the X-ray transform, well-known from computerized tomography [17].

A. X-Ray Transform

Consider the line integrals of a continuous function $f(\boldsymbol{x}), \boldsymbol{x} = (x, y, z) \in \mathbb{R}^3$, along a direction vector $\boldsymbol{\theta}$. Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be two mutually orthogonal vectors perpendicular to $\boldsymbol{\theta}$, cf. Fig. 1. The X-ray transform $\mathcal{P}_{\boldsymbol{\theta}} f$ of f is defined by

$$\begin{aligned}\mathcal{P}_{\boldsymbol{\theta}} f(u, v) &= \int_{\mathbb{R}^3} f(\boldsymbol{x})\delta(\boldsymbol{x}\cdot\boldsymbol{u} - u)\delta(\boldsymbol{x}\cdot\boldsymbol{v} - v)\, d\boldsymbol{x} \\ &= \int_{\mathbb{R}} f(u\boldsymbol{u} + v\boldsymbol{v} + t\boldsymbol{\theta})\, dt, \end{aligned} \tag{1}$$

where a dot denotes the inner product in $\mathbb{R}^3$.

The Fourier projection slice theorem [17] states that the 2-D Fourier transform of the X-ray transform $\mathcal{P}_{\boldsymbol{\theta}} f$ equals the 3-D Fourier transform of f along a slice plane through the origin and perpendicular to $\boldsymbol{\theta}$

$$\mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f(\omega_u, \omega_v) = \mathcal{F}_3 f(\omega_u \boldsymbol{u} + \omega_v \boldsymbol{v}) \tag{2}$$

where $\mathcal{F}_n f$ denotes the n-dimensional Fourier transform of a function f.

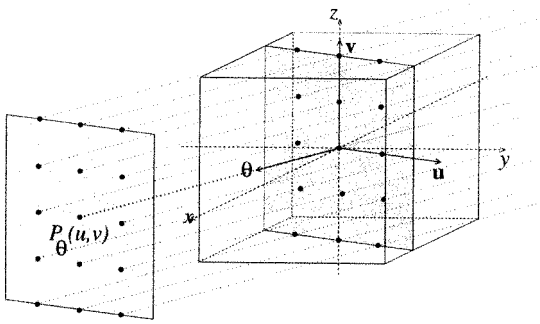

Fig. 1. View plane orthogonal to the direction vector $\boldsymbol{\theta}$.

B. FVR Algorithm

The Fourier slice theorem is the key to Fourier volume rendering. Assume that the volume data are samples on a uniform grid of a band-limited function f, whose highest frequency is determined by the sampling rate of the volume data (nonuniform grids require resampling). The FVR algorithm consists of the following steps.

- *Preprocessing*: Compute the 3-D discrete Fourier transform of the volume data by FFT.
- *Actual volume rendering*: For each direction $\boldsymbol{\theta}$, do:
  1) Interpolate the Fourier transformed data and resample on a regular grid of points in the slice plane orthogonal to $\boldsymbol{\theta}$ ("slice extraction").
  2) Compute the 2-D inverse Fourier transform, again by FFT. This yields a discrete approximation to $\mathcal{P}_{\boldsymbol{\theta}} f$.

The first step is just preprocessing: the 3-D Fourier transform is computed only once. The next two steps are repeated for each viewing direction. The time complexity for computing one view depends both on the complexity of the 2-D Fourier transform and on the interpolation cost. If the extracted slice is of size N by N, then the complexity of the Fourier transform is $\mathcal{O}(N^2 \log N)$. The complexity of 3-D interpolation is $\mathcal{O}(K^3 N^2)$, where K is the linear size of the interpolation filter with K much smaller than N. Although the Fourier transform is asymptotically dominant, in practice most of the running time is spent on interpolation.

C. Interpolation

Interpolation is the most critical step in Fourier rendering, and good interpolation functions are needed to avoid artifacts such as aliasing and dishing. Dishing is a hill-shaped weighting artifact due to the shape of the Fourier transform of the interpolation function, resulting in reduced intensities away from the center of the image. Aliasing is due to insufficient sampling. A set of discrete samples in the frequency domain corresponds to an infinitely periodic signal in the spatial domain. If the original sampling step in frequency space is F_0, one has to resample with a step size of at least $F_0/\sqrt{3}$ in order to prevent aliasing [5]. In practice one usually takes a resampling step size of $F_0/2$. A way to reduce aliasing is to pad the data in the spatial domain with zeros before the initial 3-D Fourier transform is taken. This separates the replicas in the spatial domain, and decreases the sampling distance in the frequency domain. A disadvantage of

zero-padding is increased memory usage. In the case of volume data, zero-padding by a factor of two increases the required amount of memory by a factor of eight.

Frequency domain interpolation with a filter $F(k)$ corresponds to multiplication (windowing) of the original image by the inverse Fourier transform of $F(k)$. Since ideally the window function should be rectangular, the ideal resampling filter in Fourier space is a *sinc*-function. To reduce the computational demands, interpolation functions with smaller support are used. Linear interpolation is computationally attractive, but the side lobes of the linear interpolation function in the spatial domain are the source of severe aliasing, requiring a large amount of initial zero-padding of the volume data. The cubic spline function is a better approximation of a rectangle than the linear interpolation function; it has a higher computational complexity, but requires less zero-padding [6]. The 2-D filters needed for FVR are constructed as products of the 1-D filters. Both interpolation functions have been compared by Napel *et al.* [6] for Fourier domain volume rendering. Keys [18] and Parker *et al.* [19] give a more extensive comparison of interpolation filters in general.

Malzbender [5] introduced *spatial premultiplication* as a preprocessing operation to reduce dishing. Spatial premultiplication entails point-wise multiplication of the data in the spatial domain by the reciprocal of the inverse Fourier transform of the interpolation function. Although spatial premultiplication reduces dishing, it increases the aliasing error, because effectively all periodic copies of the data in the spatial domain are premultiplied by the reciprocal function.

D. Accuracy

In order to assess the quality of various volume rendering algorithms we use a 3-D head phantom, as defined in Kak and Slaney [17], consisting of a collection of ellipsoids of different density values. Because of the linearity of the X-ray transform, a projection of an object consisting of ellipsoids is the sum of the projections of the individual ellipsoids. These projections can be computed analytically. We have slightly adapted the phantom, originally devised to test the accuracy of tomographic reconstruction algorithms, so that the range of grey values is better suited for visual inspection (see the Appendix). Besides analytical projections, we generated a volume data set of size 128^3 from the mathematical description. To reduce high frequencies in the phantom volume data, the ellipsoids are smoothed by supersampling by a factor of two and by weighting with a fourth-order B-spline filter. This makes the phantom more realistic, since data originating from a CT-scanner, for example, are low-pass filtered as well. Independent of the amount of zero padding, we use a doubling of the original sampling step in frequency space, so that volume data of size N^3 give rise to rendered images of size $(2N)^2$. Fig. 2 shows an analytical projection of the phantom.

Fig. 3 shows images of size 256^2 of the phantom data rendered by FVR without premultiplication, for both linear and cubic spline interpolation, and with 20% and 100% zero-padding. The images are displayed with the grey values inverted, so that aliasing artifacts show up more clearly. For 20% zero-padding, linear interpolation still suffers from severe

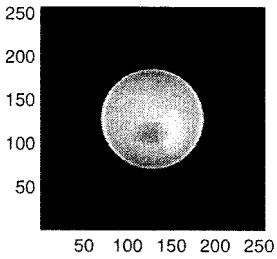

Fig. 2. Analytical projection image of the 3-D head phantom for $\boldsymbol{\theta} = (1, 0, 0)$.

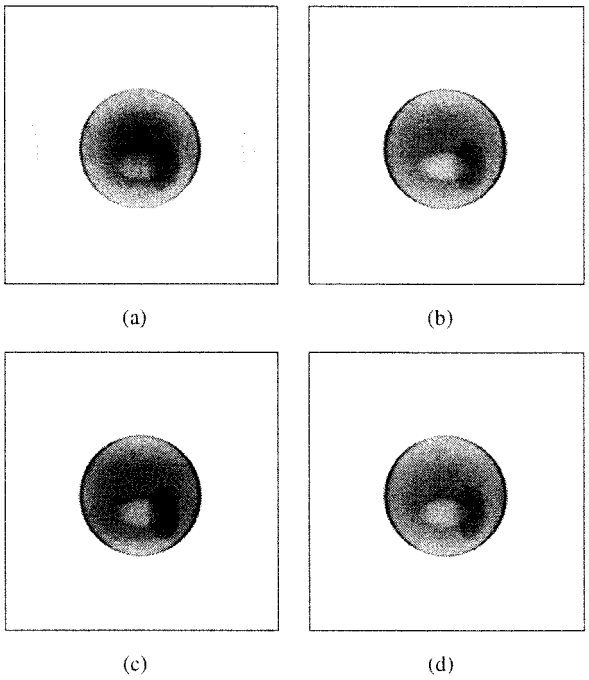

Fig. 3. Results of Fourier volume rendering with different interpolation functions and amounts of zero-padding. The images are displayed with inverted grey values. (a) Linear interpolation, 20% zero-padding. (b) Linear interpolation, 100% zero-padding. (c) Cubic spline interpolation, 20% zero-padding. (d) Cubic spline interpolation, 100% zero-padding. $\boldsymbol{\theta} = (1, 0, 0)$.

aliasing. The first replica shows up clearly, and close inspection reveals that the second replica is also visible, though very faintly, between the image in the center and the first replica near the edge. For cubic spline interpolation, aliasing is much less prominent. For 100% zero-padding, no visible aliasing occurs, but memory costs are increased by a factor of eight.

Fig. 4 shows intensity profiles corresponding to the line $x = 122$ of the images shown in Fig. 3. Without premultiplication and 20% zero-padding, dishing is very severe for linear interpolation, but almost negligible for cubic spline interpolation. Premultiplication reduces dishing, but amplifies aliasing. This explains the large values around $y = 128$ for 20% zero-padding and linear interpolation in Fig. 4(b). For both interpolation functions, dishing disappears completely with 100% zero-padding.

In conclusion, cubic spline interpolation with 20% zero-padding appears to offer a good compromise for FVR, resulting in small aliasing error and dishing artifact. Linear interpolation with 100% zero-padding produces nearly the same results as cubic spline interpolation, but memory costs

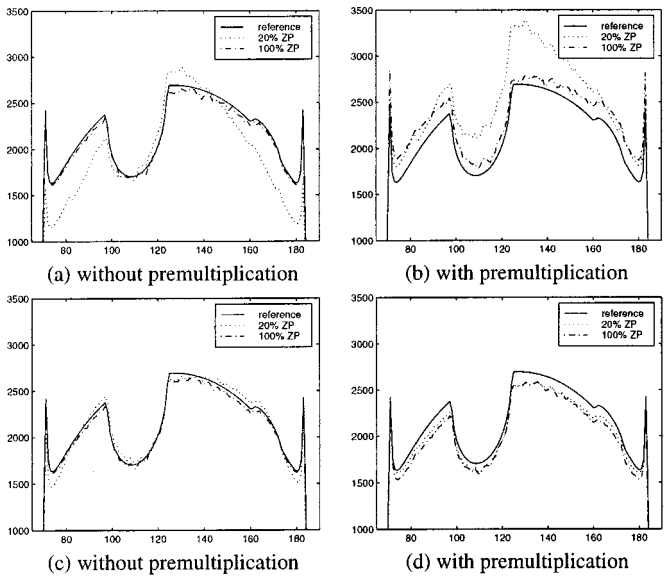

Fig. 4. Profiles of FVR renderings of the head phantom along the line $x = 122$. (a) and (b) linear interpolation, (c) and (d) cubic spline interpolation. The reference profiles are obtained by analytical computation.

are increased by a factor of eight. Premultiplication increases the aliasing error, therefore, we do not use it in this paper.

E. Reducing Memory Costs

The 3-D Fourier transform requires complex arithmetic and a floating point representation. Since the Fourier transform of a real signal is hermitian, a factor of two can be saved by dropping half of the Fourier transformed data. Whereas some authors invoke the Hartley transform for this purpose, we accomplish this by the use of a real-to-complex/complex-to-real FFT, such as provided by the package FFTW [20]. Values of coefficients which are not known directly can always be computed by complex conjugation of the corresponding coefficients in the known part of the Fourier transform. The slice extraction process makes use of this fact to save a factor of two in the number of computations.

Secondly, the original data usually has only 2 bytes per voxel. We experimentally found that, instead of using 8 bytes per voxel, it is possible to save another factor of two by quantizing the floating point values to 2-byte shorts, without seriously affecting the accuracy. This is done by scaling the floating point values linearly to the full range of shorts. During the slice extraction process, the floating point value is reconstructed, which requires one multiplication and one addition.

III. WAVELETS

In this section some basic facts about wavelet representations are introduced. In particular, we discuss the fast wavelet transform and its Fourier domain implementation.

A. Wavelet Representation

A 1-D biorthogonal wavelet basis can be constructed from a *scaling function* ϕ with associated wavelet ψ, and dual scaling function $\tilde{\phi}$ with dual wavelet $\tilde{\psi}$. The corresponding basis functions are $\{\phi_{j,k}\}$ and $\{\psi_{j,k}\}$, $j, k \in \mathbb{Z}$, where $\phi_{j,k}(x) = 2^{-j/2}\phi(2^{-j}x - k)$ and $\psi_{j,k}(x) = 2^{-j/2}\psi(2^{-j}x - k)$; the dual basis functions are defined similarly. The parameters j and k denote scale and translation, respectively. From the 1-D basis, one constructs a 2-D separable wavelet basis (of the so-called nonstandard type) with four basis functions, i.e. one scaling function $\Phi^0_{j,k,l}(x, y)$ and three wavelet basis functions $\Psi^\tau_{j,k,l}(x, y)$, $\tau \in T = \{1, 2, 3\}$, defined as follows:

$$\begin{aligned}\Phi^0_{j,k,l}(x, y) &= \phi_{j,k}(x)\phi_{j,l}(y)\\ \Psi^1_{j,k,l}(x, y) &= \phi_{j,k}(x)\psi_{j,l}(y)\\ \Psi^2_{j,k,l}(x, y) &= \psi_{j,k}(x)\phi_{j,l}(y)\\ \Psi^3_{j,k,l}(x, y) &= \psi_{j,k}(x)\psi_{j,l}(y).\end{aligned} \tag{3}$$

An analogous definition holds for the dual scaling function $\tilde{\Phi}^0_{j,k,l}(x, y)$ and wavelet basis functions $\tilde{\Psi}^\tau_{j,k,l}(x, y)$.

The M-level wavelet representation of a 2-D function f is then given by

$$\begin{aligned}f(x, y) = &\sum_{k,l} c^M_{k,l}\Phi^0_{M,k,l}(x, y)\\ &+\sum_{j=1}^{M}\sum_{\tau\in T}\sum_{k,l} d^{j,\tau}_{k,l}\Psi^\tau_{j,k,l}(x, y).\end{aligned} \tag{4}$$

The *approximation* coefficients are $c^M_{k,l} = \langle f, \tilde{\Phi}^0_{M,k,l}\rangle$ and the *detail* coefficients are $d^{j,\tau}_{k,l} = \langle f, \tilde{\Psi}^\tau_{j,k,l}\rangle$, where $\langle\cdot, \cdot\rangle$ denotes the inner product in the space $L^2(\mathbb{R}^2)$ of square integrable functions on $\mathbb{R}^2$.

In practice one deals with functions f of compact support. In order to apply the wavelet representation, f has to be extended to a function on the real line. The simplest approach, and also the one we use in this paper, is to extend f by zero values outside the support. Note that, in the following, zero-padding is not done explicitly, because the input to the wavelet transform has already been padded with zeros for slice resampling (c.f. Section II-D). The amount of zero-padding is usually large enough to prevent artifacts due to circular convolution, provided that the number of decomposition levels in the wavelet transform is kept small, and that the associated filters have a not too large support. We use this simple approach, because other methods adapt the analysis and synthesis filters near the boundaries, and are therefore difficult to use in the frequency domain.

B. Fast Wavelet Transform

The fast wavelet transform computes the wavelet decomposition, i.e., the approximation and detail coefficients, with a subband filtering scheme called the pyramid algorithm [21]. For the 1-D case, the basis functions ϕ and ψ are represented by discrete filters $h = (h_n)_{n\in\mathbb{Z}}$ and $g = (g_n)_{n\in\mathbb{Z}}$, respectively. Furthermore, there exist dual filters $\tilde{h}$ and $\tilde{g}$; for the orthogonal case these are defined by $\tilde{h}_n = \overline{h_{-n}}$ and $\tilde{g}_n = \overline{g_{-n}}$ (here $\overline{f}$ denotes the complex conjugate of f). The filters $\tilde{h}$ and $\tilde{g}$ are used in the forward wavelet transform, and are therefore called decomposition or analysis filters; the filters h and g are used for the inverse wavelet transform, and are called reconstruction or synthesis filters.

The 2-D basis (3) is represented by the four possible tensor products, hh, hg, gh and gg, of the 1-D filters h and g. [For example $(hh)_{k,l} = h_k h_l$.] Let c^j denote the 2-D sequence $(c^j_{k,l})_{k,l\in\mathbb{Z}}$, and similarly for $d^{j,\tau}$. The wavelet decomposition computes the sequences c^{j+1} and $d^{j+1,\tau}$ from c^j by convolution followed by downsampling, as follows:

$$\begin{aligned} c^{j+1} &= \downarrow_2 \left(\widetilde{hh} * c^j\right) & d^{j+1,1} &= \downarrow_2 \left(\widetilde{hg} * c^j\right) \\ d^{j+1,2} &= \downarrow_2 \left(\widetilde{gh} * c^j\right) & d^{j+1,3} &= \downarrow_2 \left(\widetilde{gg} * c^j\right) \end{aligned} \quad (5)$$

for $j = 0, \cdots, M-1$. Here $*$ denotes discrete 2-D convolution, and $\downarrow_2$ denotes downsampling by a factor of two in both dimensions. The filters $\widetilde{hh}$, $\widetilde{hg}$, $\widetilde{gh}$, and $\widetilde{gg}$ are the dual filters. [For example $(\widetilde{hh})_{k,l} = \tilde{h}_k \tilde{h}_l$.]

Wavelet reconstruction is performed recursively starting at level M by upsampling (denoted by $\uparrow\uparrow_2$) followed by convolution:

$$\begin{aligned} c^j =& hh * \left(\uparrow\uparrow_2 c^{j+1}\right) + hg * \left(\uparrow\uparrow_2 d^{j+1,1}\right) \\ &+ gh * \left(\uparrow\uparrow_2 d^{j+1,2}\right) + gg * \left(\uparrow\uparrow_2 d^{j+1,3}\right). \end{aligned} \quad (6)$$

C. The Fast Wavelet Transform in the Fourier Domain

As motivated in the introduction, we need a way to compute the fast wavelet transform (FWT) in the frequency domain. Especially when the decomposition/reconstruction filters have large support, such a Fourier domain implementation using the fast Fourier transform (FFT) is more efficient than a direct computation, as shown in [15], [16] for the 1-D case. We now consider the extension of this method to 2-D signals.

1) Up- and Down-sampling: Upsampling and downsampling can be expressed in the frequency domain using Z-transforms. The Z-transform of a 2-D discrete signal $\{x(n_1, n_2) : n_1 = 0, 1, 2, \cdots, N_1 - 1, n_2 = 0, 1, 2, \cdots, N_2 - 1\}$ is defined by

$$X(z_1, z_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2}, \qquad z_1, z_2 \in \mathbb{C}. \quad (7)$$

On the unit circle, the Z-transform $X(e^{(2\pi ik)/N_1}, e^{(2\pi il)/N_2})$ coincides with the element $X_{k,l}$ of the 2-D discrete Fourier transform (DFT) of the signal x of length N_1 by N_2. We can split this Z-transform into contributions of the samples with even and odd index, a classical technique known as *biphase decomposition* in filter bank design [15], [16]. Downsampling corresponds to taking the samples with even index in both dimensions, leading to a signal with Z-transform given by

$$\begin{aligned} X^{\text{down}}(z_1, z_2) =& \tfrac{1}{4}\left(X\left(z_1^{1/2}, z_2^{1/2}\right) + X\left(-z_1^{1/2}, z_2^{1/2}\right)\right. \\ &\left. + X\left(z_1^{1/2}, -z_2^{1/2}\right) + X\left(-z_1^{1/2}, -z_2^{1/2}\right)\right). \end{aligned}$$

The values $X^{\text{down}}_{k,l}$, $k = 0, \cdots, N_1/2-1$, $l = 0, \cdots, N_2/2-1$ of the 2-D DFT of the downsampled signal are given by

$$\begin{aligned} X^{\text{down}}_{k,l} &= X^{\text{down}}\left(e^{(2\pi ik)/(N_1/2)}, e^{(2\pi il)/(N_2/2)}\right) \\ &= \tfrac{1}{4}\left(X_{k,l} + X_{k-(N_1/2),l} + X_{k,l-(N_2/2)}\right. \\ &\quad \left. + X_{k-(N_1/2),l-(N_2/2)}\right). \end{aligned} \quad (8)$$

So, for a DFT that stores the DC component in the first array position, downsampling is implemented by dividing the 2-D array of DFT coefficients of x into two in both dimensions, and averaging the resulting four subarrays.

Conversely, upsampling by a factor of two in the spatial domain means inserting zeros between the samples in both dimensions. The Z-transform of the upsampled signal is given by

$$X^{\text{up}}(z_1, z_2) = X\left(z_1^2, z_2^2\right).$$

So, if the array of DFT coefficients has dimensions N_1 by N_2, the DFT of the upsampled signal is a $2N_1 \times 2N_2$ array obtained by replicating this array in both dimensions.

2) Wavelet Decomposition and Reconstruction: With the above results, it is possible to represent the 2-D wavelet transform in the frequency domain. Denote the Z-transforms of the filters $\widetilde{hh}$, $\widetilde{hg}$, $\widetilde{gh}$ and $\widetilde{gg}$ by $\widetilde{HH}$, $\widetilde{HG}$, $\widetilde{GH}$ and $\widetilde{GG}$, respectively. For example, $\widetilde{HH}(z_1, z_2) = \tilde{H}(z_1)\tilde{H}(z_2)$, where $\tilde{H}(z) := \sum_n \tilde{h}_n z^{-n}$ is the 1-D Z-transform of $\tilde{h}$, etc. In the same way, denote the Z-transform of c^j by $C^j(z_1, z_2)$ and that of $d^{j,\tau}$ by $D^{j,\tau}(z_1, z_2)$. Since convolution in the spatial domain is equivalent to multiplication in the Fourier domain, we can rewrite (5) in terms of Z-transforms

$$\begin{aligned} C^{j+1}(z_1, z_2) &= \left[\widetilde{HH} \cdot C^j\right]^{\text{down}}(z_1, z_2) \\ D^{j+1,1}(z_1, z_2) &= \left[\widetilde{HG} \cdot C^j\right]^{\text{down}}(z_1, z_2) \\ D^{j+1,2}(z_1, z_2) &= \left[\widetilde{GH} \cdot C^j\right]^{\text{down}}(z_1, z_2) \\ D^{j+1,3}(z_1, z_2) &= \left[\widetilde{GG} \cdot C^j\right]^{\text{down}}(z_1, z_2) \end{aligned} \quad (9)$$

where the dot denotes pointwise function multiplication.

The reconstruction (6) becomes

$$\begin{aligned} C^j(z_1, z_2) =& HH(z_1, z_2) C^{j+1}\left(z_1^2, z_2^2\right) \\ &+ HG(z_1, z_2) D^{j+1,1}\left(z_1^2, z_2^2\right) \\ &+ GH(z_1, z_2) D^{j+1,2}\left(z_1^2, z_2^2\right) \\ &+ GG(z_1, z_2) D^{j+1,3}\left(z_1^2, z_2^2\right). \end{aligned} \quad (10)$$

In practice, we deal with a *finite* 2-D input sequence c^0, represented by an array of size $N_1 \times N_2$. In the frequency domain, the result of an M-level decomposition then yields an approximation array C^M of size $2^{-M}N_1 \times 2^{-M}N_2$, and detail arrays $D^{j,\tau}$, $j = M, M-1, \cdots, 1, \tau = 1, 2, 3$ of size $2^{-j}N_1 \times 2^{-j}N_2$. An inverse FFT of C^M and of $D^{j,\tau}$ yields the arrays c^M, $d^{j,\tau}$ of the wavelet decomposition of c^0 in the original domain. However, since reconstruction also can be performed in frequency space, it is not necessary to carry out these inverse FFT's. Instead, a reconstruction at a desired level K is first computed in the Fourier domain by (10) and the resulting approximation C^K is then inversely Fourier transformed to give the desired approximation c^K.

The DFT values of the filters $\widetilde{HH}$, $\widetilde{HG}$, $\widetilde{GH}$, and $\widetilde{GG}$ are computed from $\tilde{H}$ and $\tilde{G}$ as follows. The length of the signal in the decomposition (9) decreases with increasing scale level j. At level j, let the signal length in a given spatial direction be

$2^{-j}N$, larger than the length L of the filter $\tilde{h}$. Then the required DFT values of $\tilde{H}$ at level j are

$$\tilde{H}_k^j = \tilde{H}\left(\exp\left[\frac{2\pi ik}{2^{-j}N}\right]\right) = \sum_{n=0}^{L-1} \tilde{h}_n e^{-(2\pi ink2^j/N)} = \tilde{H}_{k2^j}^0, \qquad k = 0, 1, \cdots, 2^{-j}N - 1.$$

The DFT values of $\tilde{G}$ are obtained in the same way. Clearly, it is sufficient to compute the DFT filters $\tilde{H}^0$ and $\tilde{G}^0$. The filters for the other scales are obtained by downsampling the filters for the finest scale $j = 0$. The DFT values of the synthesis filters are obtained in an analogous way. Again, it suffices to compute the filters once for $j = 0$, the filters for the other scales being obtained by downsampling. Based on the 1-D filter coefficients, we define 2-D filter matrices $\mathbf{H}^j$ and $\mathbf{G}^{j,\tau}$, $\tau = 1, 2, 3$, by

$$\left(\mathbf{H}^j\right)_{k,l} = H_k^j H_l^j, \qquad \left(\mathbf{G}^{j,1}\right)_{k,l} = H_k^j G_l^j,$$
$$\left(\mathbf{G}^{j,2}\right)_{k,l} = G_k^j H_l^j, \qquad \left(\mathbf{G}^{j,3}\right)_{k,l} = G_k^j G_l^j$$

and similarly for the dual filters. Also, C^j is the matrix with elements $C_{k,l}^j := C^j(e^{(2\pi ik)/N_1}, e^{(2\pi il)/N_2})$, with $D^{j,\tau}$ defined analogously. Then the wavelet decomposition (9) has the matrix representation

$$C^{j+1} = \left[\tilde{\mathbf{H}}^j \bullet C^j\right]^{\text{down}}, \qquad D^{j+1,\tau} = \left[\tilde{\mathbf{G}}^{j,\tau} \bullet C^j\right]^{\text{down}} \tag{11}$$

where $A \bullet B$ denotes pointwise multiplication of matrices A and B, and, for a matrix $\mathbf{X}$ with an even number of rows and columns, $\mathbf{X}^{\text{down}}$ is defined by [cf. (8)]

$$\mathbf{X}^{\text{down}} = \tfrac{1}{4}(\mathbf{X}_a + \mathbf{X}_b + \mathbf{X}_c + \mathbf{X}_d) \text{ when } \mathbf{X} = \begin{pmatrix} \mathbf{X}_a & \mathbf{X}_b \\ \mathbf{X}_c & \mathbf{X}_d \end{pmatrix}$$

where $\mathbf{X}_a$, $\mathbf{X}_b$, $\mathbf{X}_c$, $\mathbf{X}_d$ are the four submatrices obtained by dividing $\mathbf{X}$ into two along the row and column direction.

Wavelet reconstruction (10) has the matrix representation

$$C^j = \mathbf{H}^j \bullet \left[C^{j+1}\right]^{\text{up}} + \sum_{\tau=1}^{3} \mathbf{G}^{j,\tau} \bullet \left[D^{j+1,\tau}\right]^{\text{up}} \tag{12}$$

where, for any matrix $\mathbf{X}$ of Fourier coefficients, $\mathbf{X}^{\text{up}}$ is the matrix twice its size, defined by

$$\mathbf{X}^{\text{up}} = \begin{pmatrix} \mathbf{X} & \mathbf{X} \\ \mathbf{X} & \mathbf{X} \end{pmatrix}$$

We will refer to (11) and (12) as *Fourier-wavelet decomposition* (FWD) and *Fourier-wavelet reconstruction* (FWR), respectively.

For a pseudocode of wavelet decomposition and reconstruction in the Fourier domain, the reader is referred to Figs. 5 and 6, respectively. In the pseudocode, A^T denotes the transpose of A, $dsamp(A, k)$ denotes downsampling of A by a factor of k in both dimensions, and $A(p : q; r : s)$ is the submatrix of A obtained by retaining only those rows i and columns j for which $p \le i \le q$ and $r \le j \le s$.

procedure FWD
{Input: $N \times N$ matrix X,
containing 2-D FFT of input image
Output: matrix W containing FWD}

{N-point 1-D FFT of decomposition filters}
$\tilde{H} = \text{FFT}(\tilde{h}, N)$; $\tilde{G} = \text{FFT}(\tilde{g}, N)$

{compute 2-D decomposition filters}
$\tilde{\mathsf{H}} = \tilde{H}\tilde{H}^T$; $\widetilde{\mathsf{G}^1} = \tilde{H}\tilde{G}^T$
$\widetilde{\mathsf{G}^2} = \tilde{G}\tilde{H}^T$; $\widetilde{\mathsf{G}^3} = \tilde{G}\tilde{G}^T$

{pyramid algorithm, M levels}
for $j = 1$ to M **do**
{filter and decimate in frequency space}
$C = [\tilde{\mathsf{H}} \bullet X]^{\text{down}}$; $D1 = [\widetilde{\mathsf{G}^1} \bullet X]^{\text{down}}$
$D2 = [\widetilde{\mathsf{G}^2} \bullet X]^{\text{down}}$; $D3 = [\widetilde{\mathsf{G}^3} \bullet X]^{\text{down}}$
{put detail coefficients in output matrix W}
$W(1 : N/2; N/2 + 1 : N) = D1$
$W(N/2 + 1 : N; 1 : N/2) = D2$
$W(N/2 + 1 : N; N/2 + 1 : N) = D3$
{set up next iteration}
X=C
{downsample filters}
$\tilde{\mathsf{H}} = dsamp(\tilde{\mathsf{H}}, 2)$; $\widetilde{\mathsf{G}^1} = dsamp(\widetilde{\mathsf{G}^1}, 2)$
$\widetilde{\mathsf{G}^2} = dsamp(\widetilde{\mathsf{G}^2}, 2)$; $\widetilde{\mathsf{G}^3} = dsamp(\widetilde{\mathsf{G}^3}, 2)$
{reduce size}
N =N/2
end for
{put lowest approximation coefficients in W}
$W(1 : N; 1 : N) = C$

Fig. 5. Two-dimensional wavelet decomposition in the Fourier domain. $\tilde{h}$, $\tilde{g}$ are column vectors of 1-D analysis filter coefficients.

D. Time Complexity

Assume that the input image c^0 is square and contains N^2 elements, with N a power of two. In this case, the maximal number of decomposition levels is $M = \log_2 N$. The first step is a 2-D FFT of c^0 yielding an array C^0 of Fourier coefficients. We express the time complexity of the wavelet transform in the number of complex multiplications in the frequency domain. The number of multiplications for the first decomposition level is $4N^2$, since there are four filters. For the second decomposition level, both the array size and the filter lengths are reduced by a factor of two in both dimensions, so that $4(N/2)^2$ multiplications are needed; etc. Altogether, the total number of multiplications for M decomposition levels is given by

$$4\sum_{j=0}^{M-1}\left(\frac{N}{2^j}\right)^2 = \frac{16}{3}N^2(1 - 4^{-M}) \overset{M=\log_2 N}{\le} \frac{16}{3}(N^2 - 1).$$

Therefore, the time complexity of the computations in the frequency domain is $\mathcal{O}(N^2)$.

An approximation image in the spatial domain is obtained by inverse FFT of the corresponding array in the Fourier domain. Since the complexity of the initial and final FFT is $\mathcal{O}(N^2 \log_2 N)$, we conclude that the overall complexity of a Fourier domain implementation of the 2-D wavelet transform is $\mathcal{O}(N^2 \log_2 N)$. It can be shown in a similar way that the reverse transform has the same time complexity.

procedure FWR
{Input: $N \times N$ matrix X containing FWD
Output: matrix W containing FWR. A 2-D IFFT of W gives the inverse wavelet transform in the spatial domain}

{set initial blocksize}
$K = N/(2^M)$
{N-point 1-D FFT of reconstruction filters}
$H = \text{FFT}(h, N)$; $G = \text{FFT}(g, N)$
{full-size 2-D reconstruction filters}
$\mathsf{H}_{\text{fs}} = HH^T$; $\mathsf{G}^1_{\text{fs}} = HG^T$
$\mathsf{G}^2_{\text{fs}} = GH^T$; $\mathsf{G}^3_{\text{fs}} = GG^T$
{initialize W}
$W = X$

{pyramid algorithm, M levels}
for $j = 1$ to M **do**
  {downsample by factor dsf}
  $dsf = N/(2K)$
  $\mathsf{H} = dsamp\,(\mathsf{H}_{\text{fs}}, dsf)$; $\mathsf{G}^1 = dsamp\,(\mathsf{G}^1_{\text{fs}}, dsf)$
  $\mathsf{G}^2 = dsamp\,(\mathsf{G}^2_{\text{fs}}, dsf)$; $\mathsf{G}^3 = dsamp\,(\mathsf{G}^3_{\text{fs}}, dsf)$
  {approximation coefficients}
  $C = W(1:K; 1:K)$
  {detail coefficients}
  $D1 = W(1:K; K+1:2K)$
  $D2 = W(K+1:2K; 1:K)$
  $D3 = W(K+1:2K; K+1:2K)$
  {undecimate and filter}
  $W(1:2K, 1:2K) = \mathsf{H} \bullet [C]^{\text{up}} + \mathsf{G}^1 \bullet [D1]^{\text{up}} +$
  $+\mathsf{G}^2 \bullet [D2]^{\text{up}} + \mathsf{G}^3 \bullet [D3]^{\text{up}}$
  {set up next iteration: double size}
  K =2K
end for

Fig. 6. Two-dimensional wavelet reconstruction in the Fourier domain. h, g are column vectors of 1-D synthesis filter coefficients.

IV. Wavelet X-ray Transform

In this section the *wavelet X-ray transform*, as introduced in [12], is studied and an efficient implementation is derived by computing the wavelet transform in the frequency domain. This results in an algorithm which starts by computation of the Fourier transform in a slice plane, as in ordinary FVR, followed by a wavelet decomposition of the slice plane image in Fourier space.

The *wavelet X-ray transform* is defined by expanding the X-ray transform $\mathcal{P}_{\boldsymbol{\theta}} f$ of a function f in a 2-D wavelet series [cf. (4)]

$$\mathcal{P}_{\boldsymbol{\theta}} f(u, v) = \sum_{k,l} c^M_{k,l}(\boldsymbol{\theta}) \Phi^0_{M,k,l}(u, v) + \sum_{j=1}^{M} \sum_{\tau \in T} \sum_{k,l} d^{j,\tau}_{k,l}(\boldsymbol{\theta}) \Psi^{\tau}_{j,k,l}(u, v). \quad (13)$$

The coefficients $c^M_{k,l}$ and $d^{j,\tau}_{k,l}$, $\tau \in T = \{1, 2, 3\}$, now depend on the viewing direction $\boldsymbol{\theta}$.

This transform can be viewed as a close relative of the wavelet X-ray transform which combines integration over a line with a simultaneous 1-D wavelet transform along this line [13], [14]. The difference is, that we perform a 2-D wavelet transform in the plane perpendicular to the line.

Now we can state the main result, which is an extension of [12] to the biorthogonal case.

Theorem 1: The coefficients in the wavelet representation (13) for the X-ray transform of $f \in L^2(\mathbb{R}^3)$ are

$$c^M_{k,l}(\boldsymbol{\theta}) = \mathcal{F}_2^{-1}(\mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f \cdot \mathcal{F}_2 \tilde{\Phi}'^0_M)(2^j k, 2^j l) \quad (14)$$

and

$$d^{j,\tau}_{k,l}(\boldsymbol{\theta}) = \mathcal{F}_2^{-1}(\mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f \cdot \mathcal{F}_2 \tilde{\Psi}'^{\tau}_j)(2^j k, 2^j l) \quad (15)$$

where

$$\tilde{\Phi}'^0_M(u, v) = \overline{\tilde{\Phi}^0_{M,0,0}(-u, -v)}$$
$$\tilde{\Psi}'^{\tau}_j(u, v) = \overline{\tilde{\Psi}^{\tau}_{j,0,0}(-u, -v)}.$$

Proof: We only prove (15), the result (14) follows analogously. Using the Plancherel formula, we observe

$$d^{j,\tau}_{k,l}(\boldsymbol{\theta}) = \langle \mathcal{P}_{\boldsymbol{\theta}} f, \tilde{\Psi}^{\tau}_{j,k,l} \rangle = \langle \mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f, \mathcal{F}_2 \tilde{\Psi}^{\tau}_{j,k,l} \rangle = \iint \mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f(\omega_u, \omega_v) \overline{\mathcal{F}_2 \tilde{\Psi}^{\tau}_{j,k,l}(\omega_u, \omega_v)}\, d\omega_u\, d\omega_v. \quad (16)$$

Now

$$\begin{aligned}
&\overline{\mathcal{F}_2 \tilde{\Psi}^{\tau}_{j,k,l}(\omega_u, \omega_v)} \\
&= \iint e^{2\pi i(\omega_u u + \omega_v v)} \overline{\tilde{\Psi}^{\tau}_{j,k,l}(u, v)}\, du\, dv \\
&= \iint e^{2\pi i(\omega_u u + \omega_v v)} \tilde{\Psi}'^{\tau}_j(2^j k - u, 2^j l - v)\, du\, dv \\
&= e^{2\pi i(2^j k \omega_u + 2^j l \omega_v)} \iint e^{-2\pi i(\omega_u u' + \omega_v v')} \tilde{\Psi}'^{\tau}_j(u', v')\, du'\, dv' \\
&= e^{2\pi i(2^j k \omega_u + 2^j l \omega_v)} \mathcal{F}_2 \tilde{\Psi}'^{\tau}_j(\omega_u, \omega_v).
\end{aligned}$$

Using this in (16), we find

$$\begin{aligned}
d^{j,\tau}_{k,l}(\boldsymbol{\theta}) &= \iint \mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f(\omega_u, \omega_v) e^{2\pi i(2^j k \omega_u + 2^j l \omega_v)} \\
&\quad \cdot \mathcal{F}_2 \tilde{\Psi}'^{\tau}_j(\omega_u, \omega_v)\, d\omega_u\, d\omega_v \\
&= \mathcal{F}_2^{-1}(\mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f \cdot \mathcal{F}_2 \tilde{\Psi}'^{\tau}_j)(2^j k, 2^j l).
\end{aligned}$$

■

By the Fourier slice theorem (2), $\mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f(\omega_u, \omega_v) = \mathcal{F}_3 f(\omega_u \boldsymbol{u} + \omega_v \boldsymbol{v})$. Therefore, the wavelet coefficients at scale j in (13) can be computed by multiplying a slice of the 3-D Fourier transform of f by the 2-D Fourier transform of the scaling or wavelet function at scale j, followed by an inverse 2-D Fourier transform evaluated at the points of the form $(2^j k, 2^j l)$ in the view plane.

A. Algorithm

The proposed wavelet extension of FVR requires only a small modification of the standard algorithm. The implementation is facilitated by the fact that wavelet decomposition and reconstruction can be performed in the Fourier domain, see Section III-C. The algorithm, henceforth referred to as Fourier-wavelet volume rendering (FWVR), can be summarized as follows.

- *Preprocessing*. Compute the 3-D Fourier transform of the volume data (size N^3).
- *Actual volume rendering*. For each direction $\boldsymbol{\theta}$, do the following.

  1) Interpolate the Fourier transform on a regular grid of size $(2N)^2$ in the slice plane orthogonal to $\boldsymbol{\theta}$. This yields the array C^0 to be used for initializing the wavelet transform.
  2) Perform a 2-D Fourier-wavelet decomposition (FWD) of depth M, yielding approximation coefficients $C^M_{k,\boldsymbol{l}}$ and detail coefficients $D^{j,\tau}_{k,\boldsymbol{l}}$, $j = M, M-1, \cdots, 1$, respectively.
  3) Perform a partial Fourier-wavelet reconstruction (FWR) from C^M, putting all detail signals $D^{j,\tau}$ equal to zero, followed by a 2-D inverse Fourier transform to obtain an initial approximation $\hat{c}^{0,M}$ [size $(2N)^2$] in the spatial domain.
  4) Refine the approximation by partial FWR using the detail signals $D^{j,\tau}$ with $K < j \leq M$, followed by a 2-D inverse Fourier transform to obtain an approximation $\hat{c}^{0,K}$ [size $(2N)^2$] at a finer scale K in the spatial domain.

This approach enables us to implement a *client–server* visualization system. The server performs the initial 3-D Fourier transform, as well as the slicing and FWD at each view angle (steps 1 and 2), and sends the required approximation and detail coefficients to the client. The client performs the FWR and inverse Fourier transform to obtain an approximation image (steps 3 and 4). Below we describe in detail how to implement this efficiently. As long as a user is interacting with the data, only the coarsest Fourier domain approximation coefficients $C^M_{k,\boldsymbol{l}}$ are used. When interaction ceases, the Fourier domain detail coefficients $D^{j,\tau}_{k,\boldsymbol{l}}$ are taken into account, so that the client can obtain reconstructions at higher levels of detail. It is not necessary to send floating point representations (4 bytes) of the coefficients, but the coefficients can be quantized to shorts (2 bytes). The quantization error is in the order of 10^{-8}, which results in no visible artifacts.

The progressive refinement inherent in the algorithm can improve interaction with the data significantly, since the response time of the system drops. Table I provides an estimate of the time it takes to send full images of size 256^2 and 512^2 (in bytes), or approximation and detail coefficients (in shorts) only. Times are given for an ISDN connection with a bandwidth of 128 kbit/s and a 1 Mbit/s connection, which can be considered a fast Internet connection. The time for computing the wavelet decomposition is not included; also communication protocol overhead is ignored.

1) Progressive Refinement: For an M-level wavelet decomposition, progressive refinement with the pyramid algorithm is done as follows. First, an approximate reconstruction at level M in Fourier space is made by ignoring all the detail coefficients and applying a full wavelet reconstruction to the approximation coefficients. Then, a full wavelet reconstruction with the pyramid algorithm is computed with the detail coefficients of level M only, and the result is added to the approximation. This refines the approximation at level M to an approximation

TABLE I
ESTIMATED TIME (IN SECONDS) TO SEND FULL SIZED IMAGES (BYTES) OR APPROXIMATION AND DETAIL COEFFICIENTS (SHORTS) OVER AN ISDN CONNECTION AND A 1 MBIT/S CONNECTION

	ISDN (128 kbit)	Internet (1 Mbit)
256×256		
full image	4.10	0.52
approximation coefficients	0.51	0.07
detail coefficients level 2	1.54	0.20
detail coefficients level 1	6.14	0.79
512×512		
full image	16.38	2.10
approximation coefficients	0.51	0.07
detail coefficients level 3	1.54	0.20
detail coefficients level 2	6.14	0.79
detail coefficients level 1	24.55	3.15

at level $M-1$. The process continues with the detail coefficients of level $M-1$ to compute an approximation at level $M-2$, etc. That is, all images at level $M, M-1, \cdots, 0$ are successively computed.

It is possible to reduce the number of multiplications and additions by adopting the so-called *nonpyramidal* reconstruction scheme. Write the FWR equation (12) in the following form:

$$C^j = \mathcal{H}^j C^{j+1} + \sum_{\tau=1}^{3} \mathcal{G}^{j,\tau} D^{j+1,\tau}$$

where $\mathcal{H}^j$ and $\mathcal{G}^{j,\tau}$ are the operators defined by

$$\mathcal{H}^j C^{j+1} = \mathbf{H}^j \bullet [C^{j+1}]^{\mathrm{up}}, \qquad \mathcal{G}^{j,\tau} D^{j+1} = \mathbf{G}^{j,\tau} \bullet [D^{j+1,\tau}]^{\mathrm{up}}.$$

By iterating this equation, the full reconstruction C^0 can be written as follows:

$$\begin{aligned} C^0 &= \widehat{C^{0,M}} + \sum_{j=0}^{M-1} \widehat{D^{0,M}}^{-j} \\ \widehat{C^{0,M}} &= \mathcal{H}^0 \mathcal{H}^1 \cdots \mathcal{H}^{M-1} C^M \\ \widehat{D^{0,M}}^{-j} &= \mathcal{H}^0 \mathcal{H}^1 \cdots \mathcal{H}^{M-j-2} \sum_{\tau=1}^{3} \mathcal{G}^{M-j-1,\tau} D^{M-j,\tau}. \end{aligned}$$

Here, $\widehat{C^{0,M}}$ is the level M approximation reconstructed from C^M by ignoring all the detail coefficients. By successively adding full resolution images $\widehat{D^{0,M}}^{-j}$ reconstructed from $D^{M-j,\tau}$, we obtain approximations on level $M-1$, $M-2$, etc.

Since pointwise multiplication is associative, successive upsamplings implicit in the product of operators $\mathcal{H}^j$ can be performed first and combined filtering at full resolution can be done afterwards. The advantage of this approach is that the filters needed for each level can be computed in advance. This avoids all multiplications for the levels between the lowest resolution and the full resolution, needing N^2 multiplications. For the detail contributions there is an initial filtering step by $\mathcal{G}^{M-j-1,\tau}$. Since the 2-D filters are separable, only the 1-D filters for the incremental reconstruction are precomputed, which are then applied to rows and columns separately. Part of this process for the

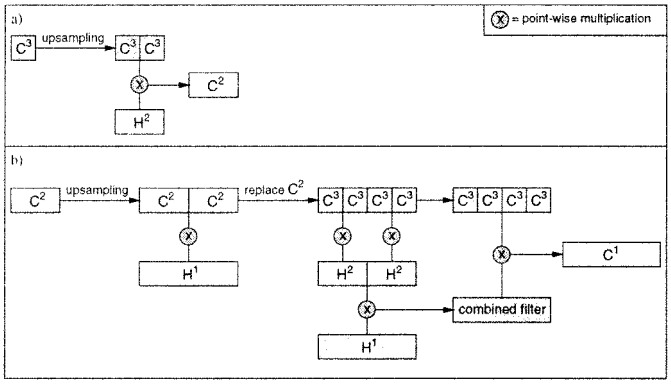

Fig. 7. (a) Computation of C^2 from C^3 via upsampling followed by multiplication. (b) Computation of C^1 from C^3 directly. The filters H^2 and H^1 are combined into a new filter, and C^3 is upsampled twice. Multiplication yields C^1.

TABLE II
COMPUTATIONAL COMPLEXITY OF PROGRESSIVE REFINEMENT IN FOURIER SPACE (IMAGE SIZE $N \times N$, M-LEVEL RECONSTRUCTION)

	pyramidal	non-pyramidal
mults	$4N^2M$	$N^2M + 4N^2(1 - 4^{-M})$
adds	$3N^2M$	$N^2M + \frac{8}{3}N^2(1 - 4^{-M})$

1-D case is illustrated in Fig. 7 for a three-level wavelet decomposition.

The time complexity of progressive refinement reconstruction to full resolution is estimated by counting the number of additions (*adds*) and multiplications (*mults*) for an M-level decomposition, see Table II (this concerns the part of the computation in the Fourier domain, i.e. excluding the final inverse 2-D FFT). From this table, it is easily verified that indeed the non-pyramidal reconstruction is more efficient than the pyramidal algorithm; equal efficiency obtains for $M = 1$. Of course the pyramidal algorithm is more efficient when only C^0 is wanted, i.e. without the intermediate approximations. Although M can be as large as $\log_2 N$, the number of decomposition levels is usually fixed to a small number, like two or three, in order for the approximation image to be useful. Therefore, the complexity of progressive refinement is $\mathcal{O}(N^2)$.

B. Results

Experiments with phantom data were carried out for two basic wavelets, the Haar wavelet and a second-order B-spline wavelet, which gives much smoother results at large compression ratios. In the latter case, we need a biorthogonal wavelet basis, defined by filters of unequal length for decomposition and reconstruction (length 41 and length 5, respectively; see [22, Appendix]). We used 20% zero-padding and cubic spline interpolation.

Fig. 8 shows the reconstruction from a three-level Haar wavelet decomposition ($M = 3$), using the approximation coefficients C^3 only, cf. Fig. 8(a), and with detail coefficients added, cf. Fig. 8(b)–(d). Fig. 9 shows the reconstruction from a three-level second-order B-spline wavelet decomposition of the same volume data. Since the number of dual coefficients of the B-spline wavelet is 41, at most three decomposition levels are possible.

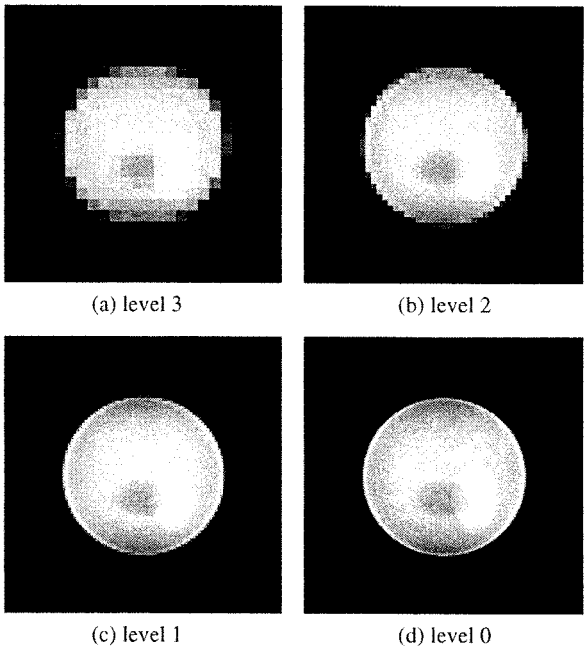

Fig. 8. FWVR rendering by a three-level Haar wavelet decomposition of phantom volume data.

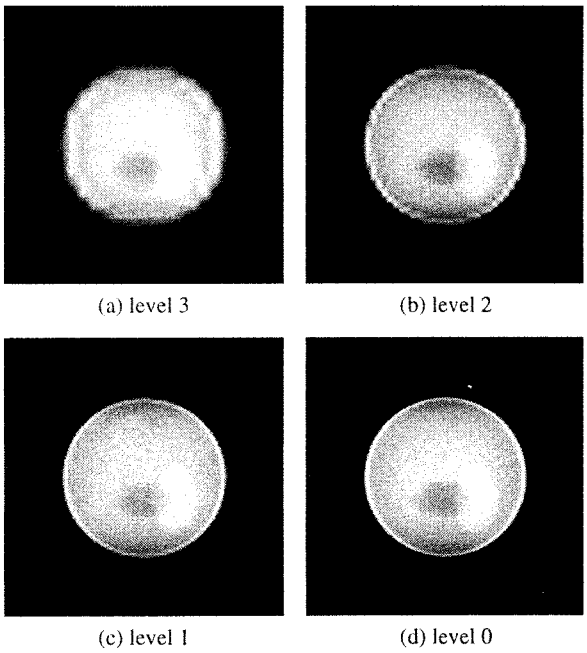

Fig. 9. FWVR rendering by a three-level second-order B-spline wavelet decomposition of phantom volume data.

Table III shows rendering times of the Fourier-wavelet rendering algorithm with progressive refinement. Linear and cubic spline interpolation with 20% zero-padding was applied. A Haar wavelet was used as a basic wavelet. Other wavelets give only marginally different timing results, in agreement with the complexity estimates in Section III-D. Three data sets were used: phantom and CT data of size 128^3, and an MR data set of size 256^3. Resolution of the slice plane was $(2N)^2$ for a dataset of

TABLE III
CUMULATIVE RENDERING TIMINGS (IN SECONDS) OF FOURIER-WAVELET VOLUME RENDERING. A HAAR WAVELET WAS USED AS A BASIC WAVELET

	Phantom 128^3	CT head 128^3	MR head 256^3
Linear interpolation			
slice extraction	0.27	0.27	1.11
FWD	0.57	0.57	2.38
Level 3 approximation			2.86
Level 2 approximation	0.68	0.68	3.64
Level 1 approximation	0.86	0.86	4.49
Full reconstruction	1.10	1.10	5.56
Cubic spline interpolation			
slice extraction	1.41	1.41	5.64
FWD	1.71	1.71	6.91
Level 3 approximation			7.40
Level 2 approximation	1.82	1.82	8.18
Level 1 approximation	2.00	2.00	9.02
Full reconstruction	2.24	2.24	10.09

size N^3. Timings were performed on a Silicon Graphics Onyx with a 200 MHz R4400 processor. Results listed for computing approximation images include the time used by the inverse 2-D FFT. The computational complexity is only dependent on the size of the rendered image, which explains the identical timings for phantom and CT data. In fact, results for these volume data are only included to facilitate a comparison with wavelet splatting below. For the same reason, the decomposition depth for the MR data was set to three levels, and for the other data sets only two levels were used. Observe that the timings increase approximately by a factor of four when the size of the data set becomes eight times as large, as expected from the theoretical complexity.

Application of progressive refinement in a client–server system is only useful if the total time needed for computing the approximation coefficients, transmitting them to the client, and performing a partial reconstruction, is smaller than the time required to send the full data over a transmission line. Referring to Table I we see that this will be the case for an ISDN line, but not for a 1 Mb/s connection (assuming this bandwidth is really available).

C. Memory Usage

The fraction k of zero-padding of the volume data determines to a large extent memory usage of Fourier rendering. Volume data are stored in a 3-D array of floats of size $(kN)^3 + (kN)^2$, where the volume data have size N^3 and k is the factor due to zero-padding. The extra $(kN)^2$ elements are required by the 3-D real-to-complex FFT. The FWD uses two temporary 2-D arrays of floats of size $(2N)^2 + 2N$. The precomputed filters for progressive refinement are 1-D. The hierarchy of combined $\mathcal{H}^j$ filters contains MN complex numbers, where M is the number of decomposition levels. The hierarchy of filters to reconstruct detail coefficients contains $2N/2^j$ complex numbers for each decomposition level j.

V. WAVELET SPLATTING

Wavelet splatting (WS) [3], [10], [11] modifies the basic splatting algorithm in two ways: i) it uses wavelets as reconstruction filters, and ii) it provides a mechanism to visualize data at different levels of detail. First, the algorithm performs a 3-D wavelet decomposition of the volume data. The 3-D separable wavelet basis with 8 basis functions is given by [c.f. (3)]

$$\begin{aligned}
\Phi^0_{j,k,l,m}(x,y,z) &= \phi_{j,k}(x)\phi_{j,l}(y)\phi_{j,m}(z)\\
\Psi^1_{j,k,l,m}(x,y,z) &= \phi_{j,k}(x)\phi_{j,l}(y)\psi_{j,m}(z)\\
\Psi^2_{j,k,l,m}(x,y,z) &= \phi_{j,k}(x)\psi_{j,l}(y)\phi_{j,m}(z)\\
\Psi^3_{j,k,l,m}(x,y,z) &= \phi_{j,k}(x)\psi_{j,l}(y)\psi_{j,m}(z)\\
\Psi^4_{j,k,l,m}(x,y,z) &= \psi_{j,k}(x)\phi_{j,l}(y)\phi_{j,m}(z)\\
\Psi^5_{j,k,l,m}(x,y,z) &= \psi_{j,k}(x)\phi_{j,l}(y)\psi_{j,m}(z)\\
\Psi^6_{j,k,l,m}(x,y,z) &= \psi_{j,k}(x)\psi_{j,l}(y)\phi_{j,m}(z)\\
\Psi^7_{j,k,l,m}(x,y,z) &= \psi_{j,k}(x)\psi_{j,l}(y)\psi_{j,m}(z)
\end{aligned}$$

Substitution of the expansion of f on this basis in (1) results in

$$\begin{aligned}
\mathcal{P}_{\boldsymbol{\theta}} f(u,v) = &\sum_{k,\boldsymbol{l},m} c^M_{k,\boldsymbol{l},m} \int_{\mathbb{R}} \Phi^0_{M,k,\boldsymbol{l},m}(u\boldsymbol{u}+v\boldsymbol{v}+t\boldsymbol{\theta})\,dt\\
&+\sum_{j=1}^{M}\sum_{\tau\in T}\sum_{k,l,m} d^{j,\tau}_{k,\boldsymbol{l},m} \int_{\mathbb{R}} \Psi^{\tau}_{j,k,\boldsymbol{l},m}(u\boldsymbol{u}+v\boldsymbol{v}+t\boldsymbol{\theta})\,dt
\end{aligned} \tag{17}$$

where $T = \{1, 2, \cdots, 7\}$. This equation expresses $\mathcal{P}_{\boldsymbol{\theta}} f(u, v)$ as a weighted summation of integrals along the line of sight. The integrals are 2-D functions on the view plane: the footprints. These have to be evaluated only once for a given viewing direction at the coarsest scale $j = M$ and translation $(k, l, m) = (0, 0, 0)$, yielding eight prototype footprints. The footprints for other scales and translations can be computed by rescaling and shifting. Prototype footprints can be computed efficiently by slicing their 3-D Fourier transforms. When analytical expressions exist for the Fourier transforms of the scaling function and wavelet, as is the case for the Haar and cardinal B-spline wavelets, no interpolation from discrete samples is necessary. For the case of B-spline wavelets, the required formulas for the Fourier transforms are somewhat involved, cf. [22, ch. 6 and Appendix]. By computing footprints analytically, we do not have problems with aliasing and dishing as in (wavelet) Fourier rendering. Also, no zero-padding of the data is necessary.

To summarize, the algorithm consists of the following steps.

1) *Preprocessing*. Perform a 3-D discrete wavelet transform (of depth M) of the volume data.
2) *Splatting*. For each viewing direction $\boldsymbol{\theta}$ do:
   - Compute prototype footprints at level M in the view plane orthogonal to $\boldsymbol{\theta}$.
   - Compute footprints for lower levels by scaling and downsampling.
   - Compute a low resolution image by summation of scaled and translated footprints weighted by the approximation coefficients $c^M_{k,l,m}$. Since, in general, the coordinates of the center of the footprint are not integers, use bilinear interpolation to convert to pixel coordinates.
   - Refine the image incrementally by adding footprints corresponding to $\Psi^{\tau}_{j,k,l,m}$, weighted by the detail coefficients $d^{j,\tau}_{k,l,m}$.

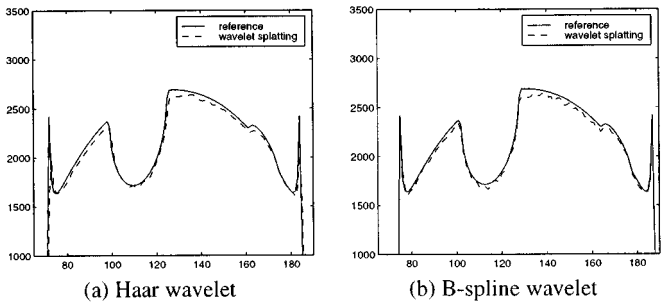

Fig. 10. Profiles along the line $x = 122$ of rendering by wavelet splatting of phantom data.

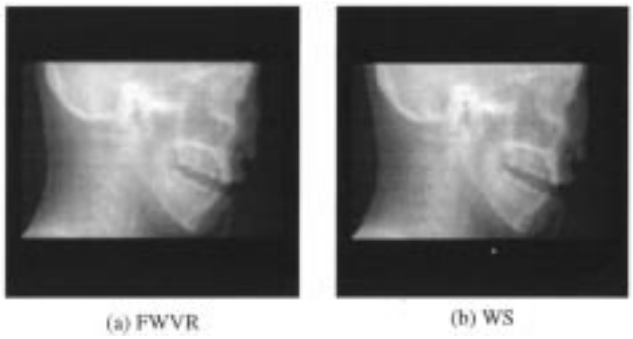

Fig. 11. Level 1 rendering of CT data (size 256^3) using a Haar wavelet.

A. Memory Usage

After the 3-D wavelet transform, the coefficients for each wavelet type and level are stored in 1-D sequences. Zero coefficients are discarded, which makes it necessary to store the spatial position of each coefficient as well. This position is encoded with an integer (4 bytes), which allows volume data of sizes up to 4 GB. Larger volumes require a different encoding for the spatial position. Storage of the position doubles memory requirements if there are no coefficients with value zero at all. In practice, a large number of data sets has 20%–40% zero coefficients, which makes memory requirements of WS comparable to those of Fourier rendering. By storing the coefficients in sequences instead of a 3-D array, the algorithm has a regular memory access pattern. This increases rendering performance, since the memory cache is fully exploited. The amount of memory used to store the footprints is dependent on the decomposition depth M and the support s of the 1-D scaling function and wavelet. The size (i.e., number of pixels) of a footprint at the coarsest scale is the first power of two larger than $s^2 2^M$. The size of the footprints decreases by a factor of four for each finer scale.

B. Results

Fig. 10 shows intensity profiles corresponding to the line $x = 122$ of rendered images at full resolution of the phantom data rendered with WS. We used a Haar wavelet and a second-order B-spline wavelet. A comparison with the plots in Fig. 4 shows that the accuracy of this method is comparable to FVR with cubic spline interpolation. Fig. 11 shows renderings of the CT data, both by FWVR and WS, also with visually very similar results.

The WS algorithm has time complexity $\mathcal{O}(N^3)$ for a $N \times N \times N$ data set, as is obvious from the summation over k, l, m in (17). Timings are shown in Table IV. Data sets and image resolution are the same as for FWVR (Table III). The scaling of the timings for increasing data size is in agreement with the theoretical complexity. Due to its larger support, the B-spline wavelet is computationally much more expensive than the Haar wavelet. Furthermore, the rendering speed is very data dependent. In case of the Haar wavelet, rendering the phantom data takes almost four times as long as rendering the CT data. The reason is that the CT data are smoother than the phantom data. For the Haar wavelet, this results in a larger number of nonzero wavelet coefficients and slower rendering, since it has only one vanishing moment. A more sparse decomposition is obtained by using B-spline wavelets which have a larger number of vanishing moments.

TABLE IV
CUMULATIVE RENDERING TIMINGS (IN SECONDS) OF WAVELET SPLATTING

	Phantom 128^3	CT head 128^3	MR head 256^3
Haar wavelet			
Level 3 approximation			1.71
Level 2 approximation	0.30	0.52	11.46
Level 1 approximation	1.15	3.49	36.89
Full reconstruction	2.59	12.62	109.81
B-spline wavelet			
Level 3 approximation			3.49
Level 2 approximation	1.04	1.07	28.99
Level 1 approximation	8.29	8.38	89.03
Full reconstruction	25.86	30.24	231.39

Comparison with Table III makes it clear that WS is indeed computationally more demanding, and also more data-dependent, than FWVR.

VI. DISCUSSION

In this paper, we have described an extension to Fourier volume rendering (FVR) based on a wavelet decomposition, which allows the data to be visualized at progressively higher levels of detail, as can be useful for client–server systems.

The *wavelet X-ray transform* was introduced, which combines integration along the line of sight with a simultaneous 2-D wavelet transform in the plane perpendicular to this line. An efficient implementation was derived by computing the wavelet transform in the frequency domain. The initial step of this Fourier-wavelet volume rendering algorithm (FWVR), i.e., computation of the Fourier transform in a slice plane, is identical to that of ordinary FVR, and requires interpolation. Aliasing can be prevented by initial zero-padding of volume data and the use of accurate interpolation filters. The additional step is a wavelet decomposition of the slice plane data in Fourier space to a given level of detail. Approximation images are then obtained by partial wavelet reconstruction in Fourier space, followed by a 2-D inverse Fourier transform. We compared the new method with wavelet splatting (WS), which modifies the basic splatting algorithm by using wavelets as reconstruction filters.

A 3-D head phantom consisting of a collection of ellipsoids of different density values was defined, for which analytical pro-

jections, as well as a volume data set of size 128^3, were computed. In the experiments, three data sets were used: phantom and CT data of size 128^3, and an MR data set of size 256^3. For a dataset of size N^3, the resolution of the slice plane was taken as $(2N)^2$. For interpolation in frequency space, we used linear and cubic spline interpolation. Experiments were carried out for two basic wavelets, the Haar wavelet and a second order B-spline wavelet. Timings were performed on a Silicon Graphics Onyx with a 200 MHz R4400 processor.

A. Differences Between FWVR and WS

In the following we enumerate the main differences between Fourier-wavelet volume rendering and wavelet splatting, and summarize the conclusions of the experimental investigations.

1) FWVR works in the frequency domain, and is initialized by a 3-D Fourier transform. WS works in the spatial domain (only the prototype footprints are obtained by Fourier domain computation), and is initialized by a 3-D wavelet transform.
2) In FWVR, a 2-D wavelet transform is computed for each view plane, therefore, the wavelet coefficients depend on the view direction. In WS, the 3-D wavelet coefficients are computed only once, and are independent of view direction.
3) The time complexity of FWVR is the same as for ordinary FVR, i.e. $\mathcal{O}(N^2 \log N)$, whereas WS has complexity $\mathcal{O}(N^3)$, just as the original splatting method.
4) To prevent aliasing, FWVR requires zero-padding of the input data. In contrast, no zero-padding is necessary in WS: footprints are computed analytically in the Fourier domain.
5) The rendering accuracy was assessed by using phantom data and comparing intensity profiles of the rendered images with analytically computed projections. Using cubic spline interpolation with 20% zero-padding we found that FWVR results in accurate renderings with quality very similar to that of WS.
6) Timings for FWVR were found to depend only on the size of the input data. In contrast, the computation time of WS is dependent on the basic wavelet used; e.g. the B-spline wavelet is computationally much more expensive than the Haar wavelet. Furthermore, the rendering speed of WS is very data dependent, because the number of nonzero wavelet coefficients depends on the smoothness of basic wavelet and volume data.
7) Memory requirements of FWVR are comparable to those of WS. FVWR needs a 3-D array of floats of size $(kN)^3 + (kN)^2$ (k is the zero-padding factor), and uses two temporary 2-D arrays of floats of size $(2N)^2 + 2N$ to compute the forward transform. In WS, the coefficients for each wavelet type and level are stored in 1-D sequences to increase rendering performance, with the spatial position of each coefficient stored as an integer. The amount of memory used to store the footprints is dependent on the decomposition depth M and the support s of the 1-D scaling function and wavelet.

TABLE V
PARAMETERS OF THE 3-D HEAD PHANTOM

Center	Axis lengths	β	Density
(0, 0, 0)	(0.69, 0.92, 0.9)	0	151.00
(0, 0, 0)	(0.6624, 0.874, 0.88)	0	-125.44
(-0.22, 0, -0.25)	(0.41, 0.16, 0.21)	108	-25.60
(0.22, 0, -0.25)	(0.31, 0.11, 0.22)	72	-25.60
(0, 0.1, -0.25)	(0.046, 0.046, 0.046)	0	25.60
(-0.08, -0.605, -0.25)	(0.046, 0.023, 0.02)	0	12.80
(0.06, -0.605, -0.25)	(0.046, 0.023, 0.02)	90	12.80
(0.06, -0.105, 0.625)	(0.056, 0.04, 0.1)	90	25.60
(0, 0.1, 0.625)	(0.056, 0.056, 0.1)	0	-25.60
(0, 0.35, -0.25)	(0.25, 0.21, 0.41)	90	25.60

B. Final Remarks

The FWVR method can be straightforwardly extended to include gradient-based shading and depth cueing [2], [3]. Also, we would like to point out that the comparison of WS and FWVR made in this paper concerns nonoptimized implementations. For example, by compression of wavelet coefficients, through thresholding or more advanced techniques, wavelet splatting can be substantially accelerated. Among such techniques we mention the embedded zerotree wavelet algorithm [23], and the conversion of wavelet transformed data into a sequential bitstream [3]. However, both methods require advanced spatial data structures, which are not necessary in FWVR.

A disadvantage of FWVR in its current form is that it requires the interpolation of a slice in Fourier space at full resolution in order to perform a 2-D wavelet decomposition. In contrast, in WS progressive refinement is immediate, since the initial data is subjected to a 3-D wavelet transform. Therefore, we plan to investigate the possibility of combining both methods to take advantage of the strengths of each of them.

APPENDIX
DEFINITION OF THE HEAD PHANTOM

The 3-D phantom used in this paper consists of a number of ellipsoids of various densities. Initially, each ellipsoid is assumed to have its three axes aligned with the axes of the coordinate system. Subsequently, a rotation of the ellipsoid around the z-axis is performed. Table V gives, for each ellipsoid, the center, lengths of the three axes of the ellipsoid, rotation angle β around the z-axis, and the density.

REFERENCES

[1] R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume rendering," *Comput. Graph.*, vol. 22, no. 4, pp. 65–74, 1988.
[2] T. Totsuka and M. Levoy, "Frequency domain volume rendering," *Comput. Graph.*, vol. 27, pp. 271–278, 1993.
[3] L. Lippert, M. H. Gross, and C. Kurmann, "Compression domain volume rendering for distributed environments," in *Proc. Eurographics'97*, 1997, pp. 95–107.
[4] F. Natterer, *The Mathematics of Computerized Tomography*. New York: Wiley, 1986.
[5] T. Malzbender, "Fourier volume rendering," *ACM Trans. Graph.*, vol. 12, no. 3, pp. 233–250, 1993.
[6] S. Napel, S. Dunne, and B. K. Rutt, "Fast Fourier projection for MR angiography," *Magn. Reson. Med.*, vol. 19, pp. 393–405, 1991.
[7] L. A. Westover, "Footprint evaluation for volume rendering," *Comput. Graph.*, vol. 24, no. 4, pp. 367–376, 1990.

[8] K. Mueller and R. Crawfis, "Eliminating popping artifacts in sheet buffer-based splatting," in *Visualization'98*, 1998, pp. 239–245.
[9] K. Mueller, T. Moeller, J. E. Swan, R. Crawfis, N. Shareef, and R. Yagel, "Splatting errors and anti-aliasing," *IEEE Trans. Vis. Comput. Graph.*, vol. 4, no. 2, pp. 178–191, 1998.
[10] L. Lippert and M. H. Gross, "Fast wavelet based volume rendering by accumulation of transparent texture maps," *Comput. Graph. Forum*, vol. 14, no. 3, pp. 431–443, 1995.
[11] L. Lippert, "Wavelet-based volume rendering," Ph.D. dissertation, Swiss Federal Institute of Technology, Zürich, 1998.
[12] J. B. T. M. Roerdink and M. A. Westenberg, "Wavelet-based volume visualization," *Nieuw Arch. Wiskunde*, vol. 17, no. 2, pp. 149–158, 1999.
[13] A. L. Warrick and P. A. Delaney, "A wavelet localized Radon transform," *Proc. SPIE*, vol. 2569, pp. 632–643, Sept. 1995.
[14] R. A. Zuidwijk, "The wavelet X-ray transform," Centre Math. Computer. Sci., Amsterdam, The Netherlands, Tech. Rep. PNA-R9703, Mar. 1997.
[15] O. Rioul and P. Duhamel, "Fast algorithms for discrete and continuous wavelet transforms," *IEEE Trans. Inform. Theory*, vol. 38, no. 2, pp. 569–586, 1992.
[16] M. Vetterli and C. Herley, "Wavelets and filter banks: Theory and design," *IEEE Trans. Signal Processing*, vol. 40, pp. 2207–2232, Sept. 1992.
[17] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. New York: IEEE Press, 1988.
[18] R. G. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 1153–1160, June 1981.
[19] J. A. Parker, R. V. Kenyon, and D. E. Troxel, "Comparison of interpolation methods for image resampling," *IEEE Trans. Med. Imag.*, vol. MI-2, no. 1, pp. 31–39, 1983.
[20] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT," in *Proc. ICASSP*, 1998, p. 1381.
[21] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, 1989.
[22] C. K. Chui, *An Introduction to Wavelets*. New York: Academic, 1992.
[23] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.

Michel A. Westenberg (S'00) received the M.Sc. degree in computing science from the University of Groningen, The Netherlands, in 1996. He is currently pursuing the Ph.D. degree in computing science at the same university.

His research interests include scientific visualization, wavelets, and biomedical image processing.

Jos B. T. M. Roerdink (M'96) received the M.Sc. degree in theoretical physics from the University of Nijmegen, The Netherlands, in 1979 and the Ph.D. degree from the University of Utrecht, The Netherlands, in 1983.

He was a Postdoctoral Fellow with the University of California, San Diego, from 1983 to 1985. He was with the Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, from 1986 to 1992, where he worked on image processing and tomographic reconstruction. He is currently an Associate Professor of computing science with the University of Groningen, The Netherlands. His current research interests include mathematical morphology, wavelets, biomedical image processing, and scientific visualization.